

Software Engineering

Books or notes are **not** allowed.

Write only on these sheets. **Concise** and **readable** answers please.

Surname, name, matricola _____

City bicycle rental system

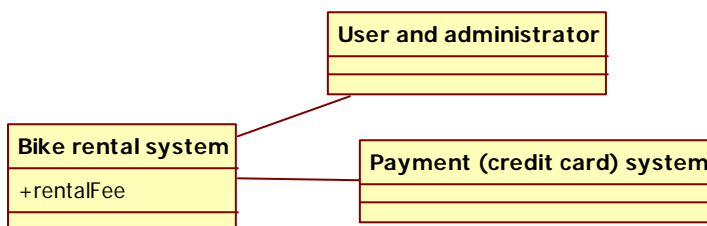
Many cities have deployed a bicycle rental system. The system is composed of many deposits, distributed all over the city. A deposit contains some dozen bicycles in an open area, the deposit has a number of numbered places (one per bicycle) where bicycles are initially stored with a lock/unlock system. The lock/unlock system is connected to the system and works as follows. If a user is authorized by the system, the system opens a lock for a bicycle, the user can get it and becomes responsible for the bicycle until it returns it. When the user wants to return a bicycle, he selects an empty place in a deposit, and inserts the bicycle in it. The lock/unlock system senses the bicycle and automatically locks the bicycle. From this moment the user is not responsible anymore for the bicycle.

To be able to use bicycles a person must first register with the system, providing his/her name and his credit card information, and obtaining an ID. Next, when a user wants to take a bicycle, he goes to a deposit, inserts in a dedicated interface (made of keyboard and screen) his ID. After the needed checks, the system selects a bicycle available in the deposit and opens the corresponding lock, so the user can take the bicycle. When the user wants to return the bicycle, he selects an empty place in a deposit, and returns the bicycle. No interaction with the keyboard/screen should be needed for return.

The rental system must track the state of all bicycles and rentals. Notably, the user pays for the rental a fee that depends on the duration of the rental. Users are encouraged to take the bicycle in one deposit and return it to any other deposit.

The system should also monitor the maintenance of bicycles (a bicycle never rented is probably broken), the distribution of bicycles in deposits (no deposit should be always empty, no deposit should be always full), the most common paths used (where bicycles are most frequently taken and most frequently returned)

1 (13 points) – a. Define the context diagram (including relevant interfaces)



Interface1: internet connection to credit card circuit (payment system). Function called: verifyCredit Card(credit card number), debit(amount). Data exchanged: credit card number, amount to be debited, result or errors (credit card invalid, credit card blocked, amount not available etc)

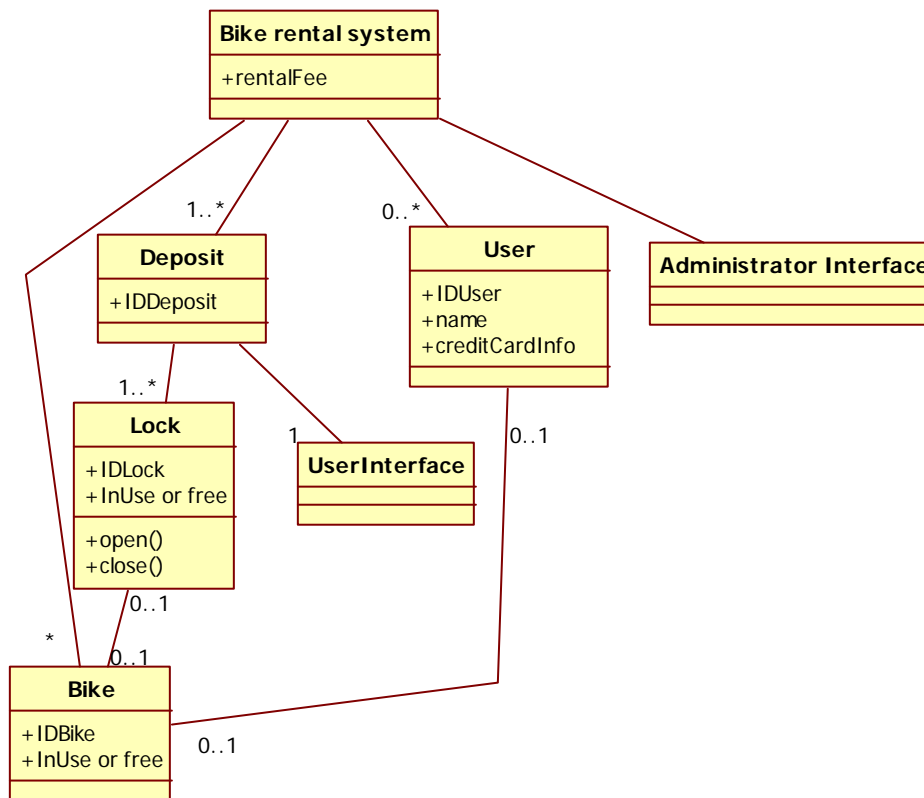
Remark that bikes, locks etc are inside the system.

Interface2: GUI to user

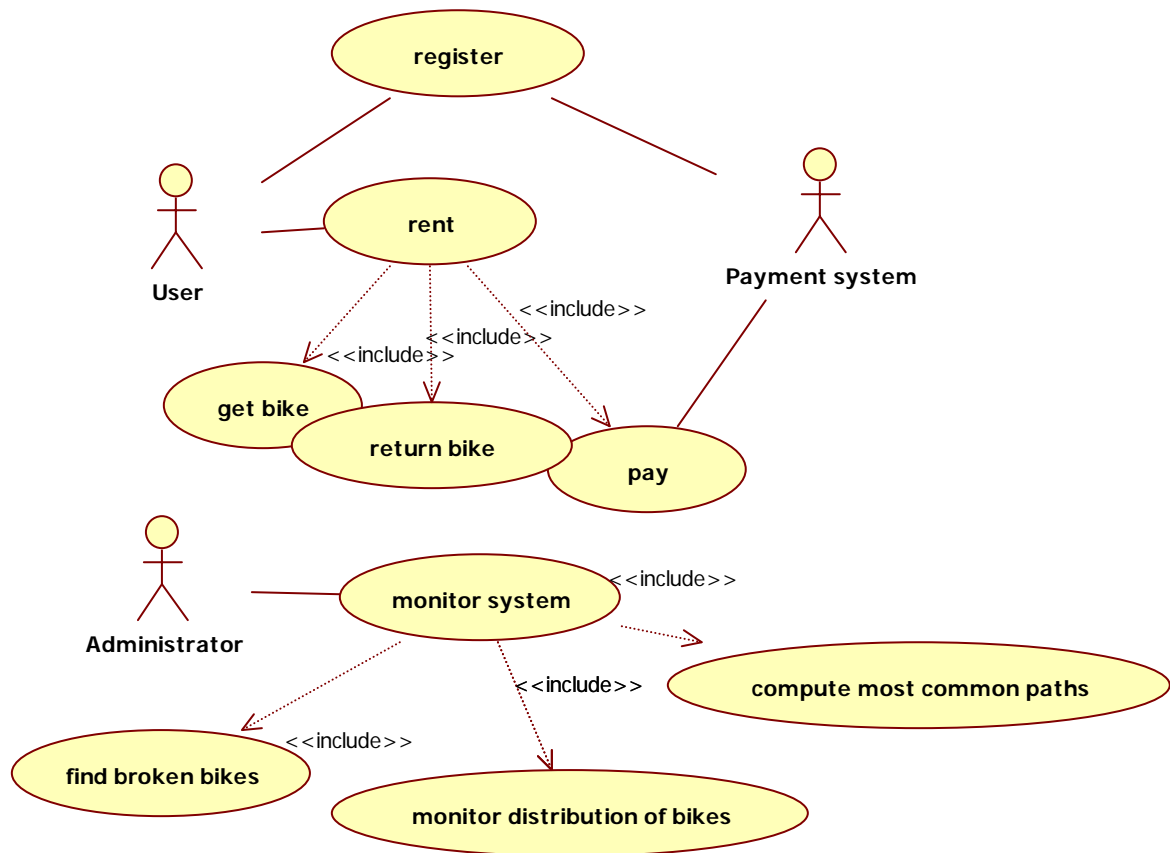
Interface3: GUI to administrator

Define the class diagram

User is the class that represents a user inside the system. User Interface represents the very simple keyboard/screen in each deposit, where the user interacts with the system. Lock represents both a place for a bicycle and the lock in this place, so a closed lock means also that the bicycle is in that position. The Administrator interface (one for the whole system) allows to monitor the system.



Define the use case diagram



Define one scenario describing a successful rental operation

Precondition: user is registered, user can pay

Postcondition: user receives a bike, system records user u got bike b

Step	Description
1	User inserts his ID in userInterface
2	System checks ID, user is registered
3	Systems checks credit, user can pay
4	System selects one available bike (bike b, in position p) and tells to user to take bike in position p
5	System opens lock in position p
6	System records that user u got bike b

Define one scenario describing an unsuccessful rental operation

Precondition: user is registered, but credit not

Postcondition:

Step	Description
1	User inserts his ID in userInterface
2	System checks ID, user is registered
3	Systems checks credit, user can't pay
4	System tells to user rental is not possible

2 (8 points) -Define black box tests for the following function

```
int bicycleRentalFee(int rentalProgram, int initTime, int endTime, int nRentals);
```

The function is an example on how the fee for bicycle rental could be computed.

initTime is the time when the bicycle was rented, endTime when it was returned. The unit is expressed in integers.

nRentals is the total number of rentals made

rentalProgram should be 1 (normal user) or 2 (frequentUser).

If 1 the fee is proportional to the time. It is computed as 2Y per unit of time

If 2 the fee is discounted. It is 0 for the first 10 units of time, 2Y per unit from 11 to 100 units of time, 1Y from 101 units and more. Besides, every 10 rentals (parameter nRentals) the rental is free.

Criterion	Valid class	Invalid class	Boundary Condition
Rental program	1,2	All other values	0, 3
initTime	0 to maxint	<0	-1, 0, 1
endTime	0 to maxint	<0	-1, 0, 1
nRentals	0 to maxint Only when rentalProgram=2, otherwise should not be used	<0	
Fee, rentalProgram = 1	(endTime - initTime) >=0	endTime < initTime	initTime = endTime
Fee, rentalProgram = 2 and nRentals mod 11 <> 0	(endTime - initTime) >=0 endTime - initTime >=0 and <=10 endTime - initTime >10 and <=100 endTime - initTime >101	endTime < initTime	initTime = endTime
Fee, rentalProgram = 2 and nRentals mod 11 == 0	(endTime - initTime) >=0	endTime < initTime	initTime = endTime

Actual test cases not reported.

3-1 (5 points) – For the following function define the control flow graph, and define test cases to obtain the highest possible node coverage, edge coverage, multiple condition coverage. Discuss the feasibility of path coverage. For the test cases, write only the input value.

Report here the coverage obtained

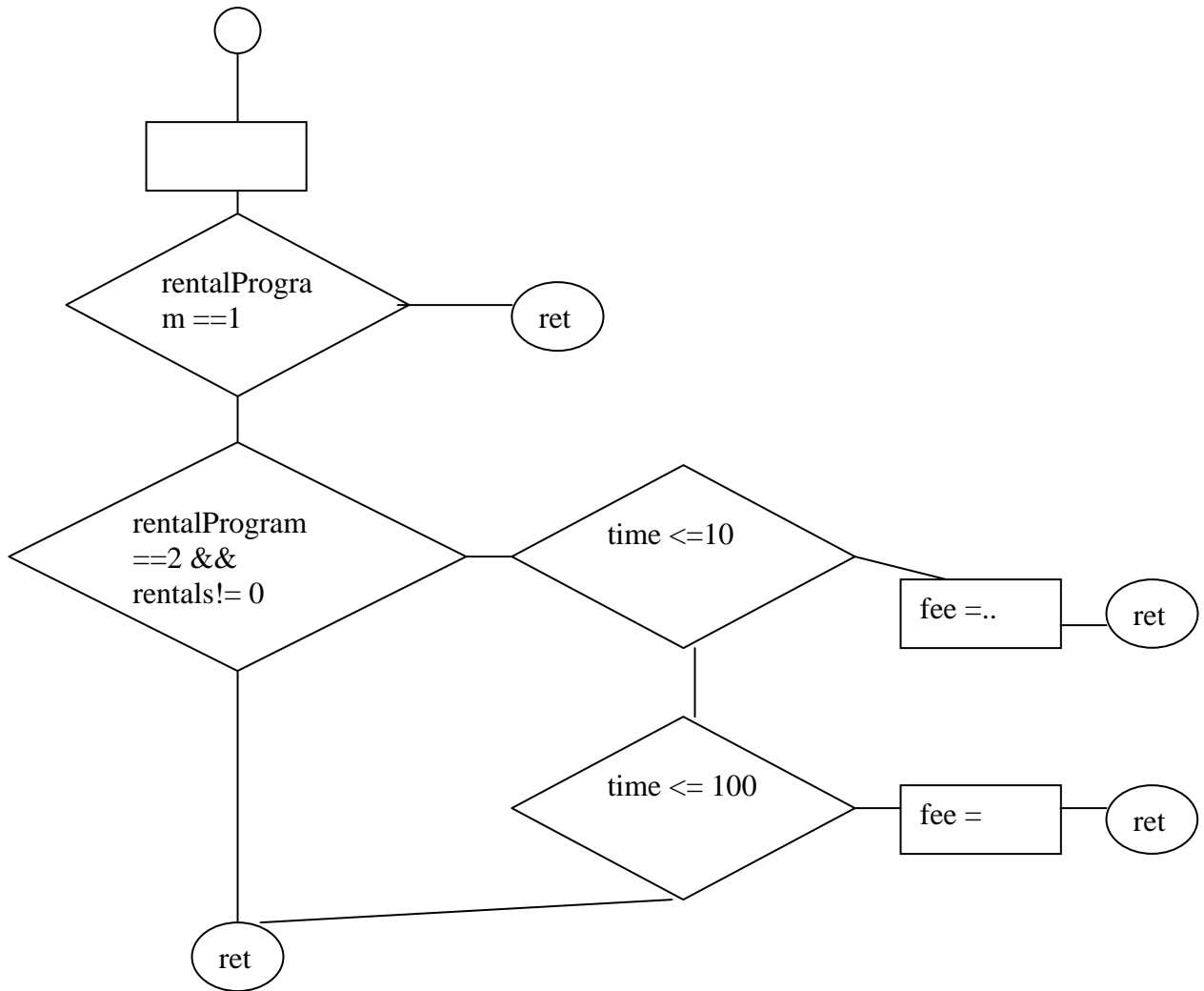
--node coverage: 100% can be achieved with 4 tests

--edge coverage: 100% can be achieved, with same 5 tests

--multiple condition coverage – this applies to `if (rentalProgram ==2 && rentals != 0)` where the condition has two terms, so in the most general case 4 test cases are needed to cover all possible combinations. This requires 2 new test cases

--path coverage - feasible because there are no loops, the number of possible paths is 5, the same test suite that gives 100% edge coverage gives 100% path coverage. Remark that path coverage in this case is less powerful than multiple condition coverage

```
int bicycleRentalFee(int rentalProgram, int initTime, int endTime, int
nRentals){
    int unitRate = 2;
    int rentals = nRentals %10;
    int time = endTime-initTime;
    int fee = 0;
    if (rentalProgram == 1){
        return time * unitRate;
    }
    else if (rentalProgram ==2 && rentals != 0){
        if (time <= 10)
            { fee = unitRate* time;
              return fee;}
        if (time <= 100)
            { fee = 10*unitRate + (time-10)* unitRate/2;
              return fee;}
    }
    return 0;
}
```



Node coverage: T1(1, 0, 1, 0)

T2(2, 1, 9, 1)

T3(2, 1, 19, 1)

T4(2, 1, 9, 0)

Edge coverage: same as above plus T5(2, 1, 109, 1)

Condition coverage: **rentalProgram == 2 && rentals != 0**

T2 or T3 or T5 cover T T

T4 covers T F

T6(3, , , 1) covers F T

T7(3, , , 0) covers F F

Path coverage: same as edge coverage

3-2 (1 points) – For the following function define tests to obtain the highest possible loop coverage

```
int divide(int number){  
    int primes[]= {1,2,3,5,7,11};  
    double f;  
    for (int i=0; i<6 ; i++){  
        f += number% primes[i];  
    }  
}
```

The for loop cannot be controlled via the input, so loop coverage will be 1/3 with any test suite of one or more test cases.

4) (1 point) Describe the bottom up test integration strategy

See slides

5) (1 points) What is an inspection? How does it work?

See slides

6) (1 points) The event ‘compilation of file A ended’ could be a milestone? ‘all integration integration test passed with no failures’ could be a milestone? Explain the rationale of your answers.

The former no (too low level an event, with no likely influence on next events), the latter yes (important event with influence on what could happen next)

7) (1 point) A project is estimated to require 12 person months effort. What could be its duration?

It depends on the number of people available. If only one person works, it will be 12 calendar months. If more people work, it will be less. For instance with two people the duration could be 6 calendar months. However, there is a limit on the amount of work that could be done in parallel. So it is unreasonable that, having 24 people, the project could end in half a calendar month

8) (1 point) What is a configuration item? And a configuration?

See slides

9) (1 point) In the context of verification and validation, describe the data flow analysis technique

See slides